RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
LEHRSTUHL FÜR TECHNISCHE INFORMATIK
Prof. Dr.-Ing. Karl-Friedrich Kraiss

STUDIENARBEIT

zum Thema

HOMOTOPIE-ERHALTENDE SKELETONS

von
cand. Ing. Daniel Ruijters
Matr.-Nr.: 195254

Gutachter:
Prof. Dr.-Ing. Karl-Friedrich Kraiss

Betreuer:
Dipl.-Ing. Pablo Alvarado

Aachen, den 4. Oktober 2000

Hiermit versichere ich, daß ich die vorliegende Studienarbeit selbständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen sind, sind als solche kenntlich gemacht.

Aachen, den 4. Oktober 2000

(Daniel Ruijters)

Kurzfassung

In dieser Arbeit wurde ein Ansatz zur homotopie-erhaltende Skeletierung von Binärbildern untersucht, bei dem morphologischen Operatoren benutzt wurden. Der gewählte Algorithmus basiert auf einer Arbeit von Liang Ji und Jim Piper [Ji und Piper, 1992]. Im Rahmen des AXIOM-Projektes wurde der Algorithmus für die Benutzung mit Binärbilder in C++ implementiert.

Abstract

The objective of this report is to present an approach for extracting homotopy-preserving skeletons out of binar images by using morphological operators. The surveyed algorithmn is based on contributions by Liang Li and Jim Piper [Ji und Piper, 1992]. As part of the AXIOM project, the considerations of Liang Li and Jim Piper have been adapted for the use with binar images and were implemented in C++ .

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das Projekt AXIOM (**A**daptive E**x**pert System for **I**ntelligent **O**bject **M**ining) hat die optische Erkennung großer Mengen dreidimensionaler Objekte zum Ziel. Als Unterscheidungskriterium sollen lokale und globale optische Merkmale der zu erkennenden Objekte dienen, die dem System durch Kameraaufnahmen zugeführt werden. Für den Erkennungsprozeß werden trainierbare Klassifikatoren verwendet, die in der Trainingsphase die präsentierten Objekte aufgrund der extrahierten Merkmale diskriminieren.

Skeletons beinhalten fundamentale Informationen über die Form eines Objektes. Ihre Generierung setzt Kenntnisse über die Zugehörigkeit eines Pixels zum Objekt oder zum Hintergrund voraus. In einer weiteren Stufe kann die Skeleton-Darstellung zur Extraktion von Formmerkmalen Anwendung finden. Skeleton Algorithmen sind in der Regel sehr Zeitaufwendig.

In dieser Arbeit wird ein alternativer, von Liang Ji und Jim Piper vorgeschlagener Ansatz untersucht, mit dem das Skeleton mit Hilfe morphologischer Operatoren extrahiert wird. Dieses Verfahren ist wesentlich schneller als klassische Skeleton Algorithmen, die auf die Medial Axis Transformation basieren [Ji und Piper, 1992] [Sonka, Hlavac und Boyle, 1998].

Im folgendem Kapitel werden die Motivation und die Grundlagen für den gewählten Ansatz erläutert. In Kapitel 3 wird der Algorithmus von Liang Ji und Jim Piper im Detail beschrieben, in Kapitel 4 ist die am Lehrstuhl entstandene C++ Implementierung mit allen wichtigen Schnittstellen und Modifikationen dokumentiert. Das fünfte Kapitel enthält Testergebnisse für ausgewählte Objekte und eine Diskussion der Resultate. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick auf weitere Entwicklungen, sowie mit einem Anhang mit Beweisen und Informationen für Programmierer.

2 Grundlagen

2.1 Formmerkmalen

In der Objekterkennung ist es von großen Bedeutung Formmerkmalen aus Objektaufnahmen zu extrahieren. Da die Skeletons wesentliche Informationen über die Form eines Objektes kennzeichnen, können sie als Grundlage für die Merkmalsberechnung dienen. Dabei ist es von großen Vorteil für die Objekterkennung wenn die Homotopie eines Objektes nicht verloren geht.

2.1.1 Objekt

Der Begriff Objekt wird hier definiert als eine Versammlung von Vordergrundpunkten die 8-Zusammenhängend sind. Das heißt: es gibt zu jedes zum Objekt gehörendes Pixel p mindestens ein anderes zum Objekt gehörendes Pixel q, daß 8-Verbunden mit p ist. $q \in N_8(p)$. Kenntnisse über die Zugehörigkeit eines Pixels zum Vorder- oder zum Hintergrund wird vorausgesetzt. Dabei ist:

$$N_4 \in \{(x+1, y), (x-1, y), (x, y+1), (x, y-1)\}$$
$$N_8 \in N_4 \cup \{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)\}$$

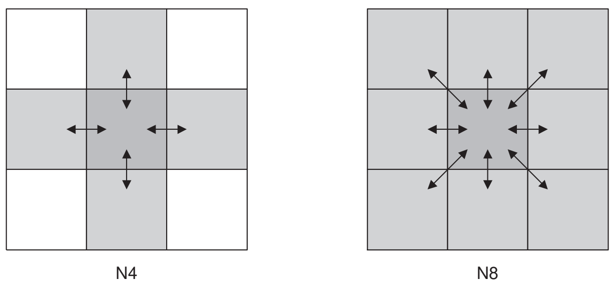

Abbildung 2.1: N4 und N8 Nachbarn

2.1.2 Homotopie Erhaltung

Eine Transformation heißt homotopie-erhaltend wenn sie nicht die kontinuierliche Beziehung zwischen verschiedene Regionen ändert. Diese Beziehung wird ausgedruckt durch den Homotopiebaum; die Wurzel korrespondiert mit dem Hintergrund, die Blätter auf der erste Stufe mit die Objekte, die Blätter auf der zweite Stufe mit den Löcher innerhalb die Objekte, usw. [Sonka, Hlavac und Boyle, 1998]

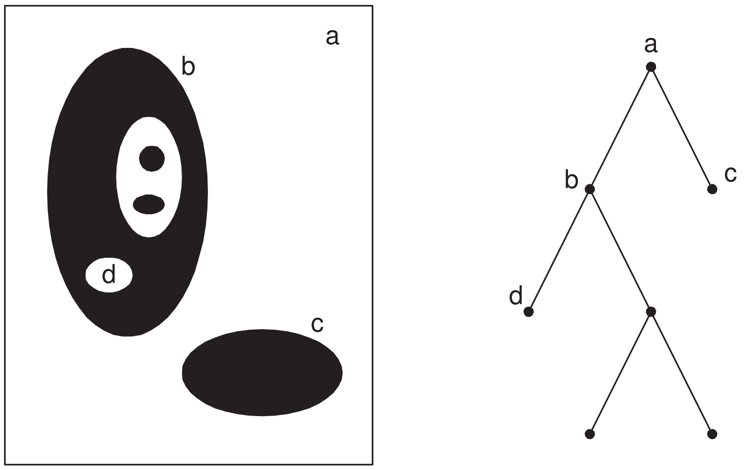

Abbildung 2.2: Objekte und ihre Homotopiebaum

2.1.3 Skeletons

Der Begriff "Skeleton" ist nicht eindeutig festgelegt. Begriffe wie "thinning", "Medial Axis (MA)" und "Skeleton" werden alle in der Literatur benutzt um die gleiche Gruppe von Algorithmen zu beschreiben. Die Grundlegende Bezeichnung eines Skeletons ist: eine Versammlung von Punkten, wobei für jeden Punkt p des Skeletons mindestens zwei Kantenpunkten eines Objektes existieren mit den der Punkt p äquidistant ist. In diesem Arbeit wird diese Definition weiter als "Formpunkte" bezeichnet. Für Skeletons wird hier gehandhabt, daß sie zusätzlich auch noch die Homotopie eines Bildes erhalten müssen. Eine wesentliche Eigenschaft von Skeletons ist das sie aus dünne Linien aufgebaut ist. Ob diese Linien nur ein Pixel breit sein dürfen oder nicht kommt auf die Anwendung und den Skeletonalgorithmus an.

2.2 Morphologie

Skeleton Algorithmen sind dafür bekannt, daß sie relativ langsam sind. Laut Literatur sind die Verfahren die auf morphologische Operationen beruhen wesentlich schneller als traditionelle Algorithmen [Ji und Piper, 1992].

2.2.1 Morphologische Operationen: Dilatation und Erosion

Die zwei grundlegende Operationen in der Morphologie sind die Dilatation und die Erosion. Bei beide Operationen wird ein Kernel und ein Bild benutzt. Der Kernel, auch "Structuring Element" genannt, wirkt dabei auf die Objekte die im Bild erhalten sind. Seine Dimensionen sind in der Regel kleiner als die des Bildes. Einige Kernels werden im Abschnitt 3.1.3 vorgestellt. Die Dilatation und Erosion basieren auf die Vektor-Addition [Sonka, Hlavac und Boyle, 1998], oder Minowski Satz Addition: $(a, b) + (c, d) = (a + c, b + d)$. Beide morphologische Operationen sind nicht reversibel.
Bei der Dilatation $\oplus$ wird das Bild X und den Kernel B mittels Vektor-Addition verknüpft. Die Dilatation $X \oplus B$ ist der Satz von alle mögliche Vektor Additionen von Paare von Elemente aus X und B:

$$X \oplus B = \{p \in \varepsilon^2 : p = x + b, x \in X, b \in B\}$$

Nach die Operation sind die Objekte im Bild X größer geworden. ε^2 steht für den euklidischen 2D Raum.

Bei der morphologische Erosion $\ominus$ wird das Bild X und den Kernel B mittels Vektor Subtraktion verknüpft. Die Erosion ist die duale Operation von der Dilatation. Nach die Operation sind die Objekte im Bild X kleiner geworden.

$$X \ominus B = \{p \in \varepsilon^2 : p + b \in X \ \forall b \in B\}$$

2.2.2 Differenz

Wenn A und B zwei Binärbilder sind, dann wird der Differenz $I = A \backslash B$ definiert als $p \in I$ wenn ($p \in A$ und $p \notin B$). Man kann sich also das Bild I vorstellen als A mit B heraus gestanzt.

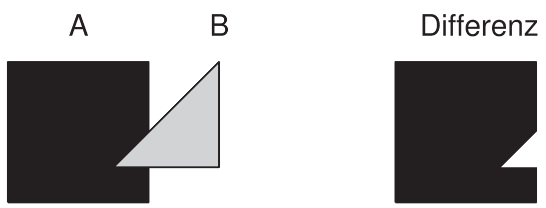

Abbildung 2.3: Differenz A\B zweier Figuren A und B

2.2.3 Distanz Transformation (DT)

Sei d die Distanz Funktion [Ji und Piper, 1992], daß heißt $d(p, q) > 0, p \neq q$ für alle $p, q \in Z^2$ und $d(p, p) = 0$ für alle $p \in Z^2$. d ist definiert als die Summe von "Nachbardistanzen" eines

Punktes und seiner N_8 Nachbarn. $d(p,q)$ ist dann der kürzeste Weg zwischen p und q. Wobei ein Weg eine Sequenz von benachbarten Pixeln ist, und seine Länge die Summe von Nachbardistanzen vom jeden Punkt zum Nächsten. Die Nachbardistanz braucht nicht unbedingt die metrische Distanz zu entsprechen, weil sie abhängig ist vom benutzten Kernel. In Abbildung 2.4 sind einige Beispiele gegeben für verschiedene Kernels und ihre Nachbardistanzen. Die Distanz Transformation (DT) eines Punktes p in einem Bild I sei:

$$D(p) = min(d(p,v)|v \in \overline{I})$$

v ist ein Punkt in das Komplement von I, also ein Hintergrundpunkt. Also ist $D(p)$ der kleinste Abstand von p zum Hintergrund.
Ein Punkt p ist ein lokales Maximum genau dann wenn $D(p) + d(p,q) > D(q)$ für alle $p \neq q$. Der Satz von lokale Maxima entspricht die "Formpunkte".

2	1	2
1	0	1
2	1	2

Cityblock (N4) Kernel

1	1	1
1	0	1
1	1	1

Chessboard (N8) Kernel

Abbildung 2.4: Die Nachbardistanzen für zwei Kernels

2.2.4 Skeletons mittels Morphologie

E sei eine Sequenz von Kernels. Jedes Element E_i ist gekennzeichnet durch seinem Durchmesser $\rho = (i * 2) - 1$. Also hat das kleinste Element E_1 einen Durchmesser von $\rho = 3$, das nächste $\rho = 5$ usw. I ist ein Bild das ein oder mehrere Objekte enthält. Jetzt sei I_i das Bild das mit E_i erodiert wurde, $I_i = I \ominus E_i$. Die i-te Hülle von das Orginalbild ist dann $S_i = I_i \backslash I_{i+1}$. Also wenn man sich die Sequenz von I_i anschaut, wird immer eine Hülle von 1 Pixel breit von die Objekte in I entfernt. Bewertet man jede Hülle S_i mit i, dann erhält man die Distanz Transformation (DT) von Bild I.
Die Differenz von I_i und die Dilatation von I_{i+1} mit ein Kernel D liefert die lokale Maxima M_i gehörend zu die Hülle S_i. Also $M_i = I_i \backslash (I_{i+1} \oplus D)$. Die Sequenz von alle M_i entspricht also die "Formpunkte". Das Skeleton muß jetzt nur noch ergänzt werden mit die homotopieerhaltende Punkte, auch J-Punkte genannt.
Wenn E eine trennbare Sequenz von Kernels ist, kann I_i viel effektiver berechnet werden [Ji und Piper, 1992]. E ist eine trennbare Sequenz, wenn gilt $E_{i+1} = E_i \oplus E_1$, also wenn man das nächste Element in der Sequenz berechnen kann durch eine Dilatation von das letzte Element mit dem ersten Kernel. In diesen Fall ist $I_{i+1} = I_i \ominus E_1$. Man erhält einen wesentlichen Geschwindigkeitsgewinn weil man in jedem Iterationsschritt nur mit dem kleinsten Kernel erodiert, statt mit einem immer wachsenden Kernel.

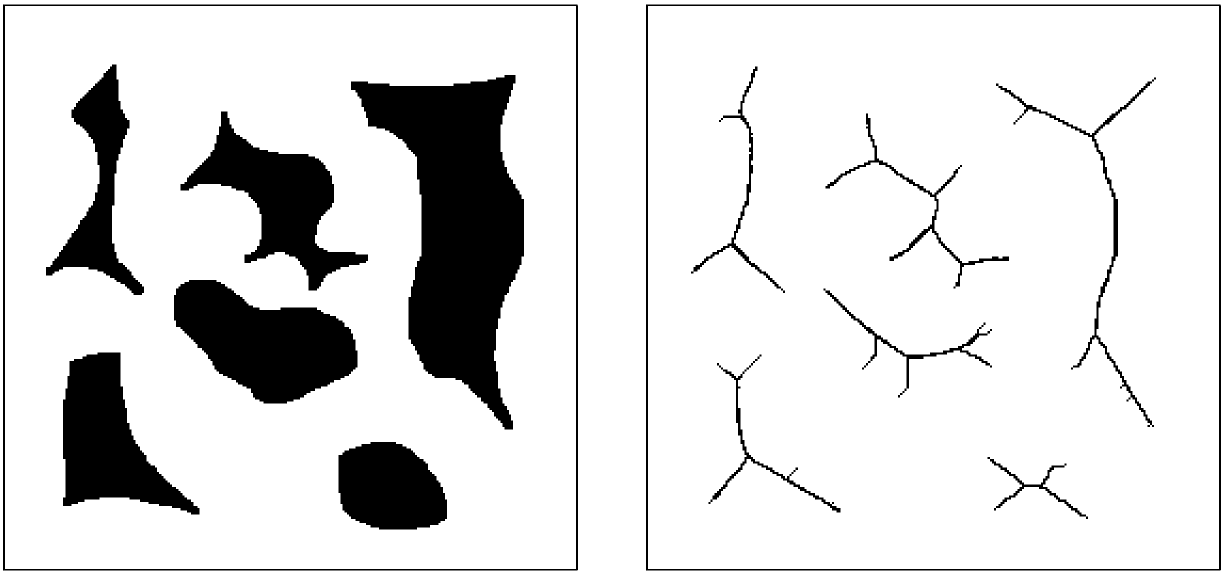

Abbildung 2.5: Einige Objekte und ihre Skeletons

3 Der Algorithmus von Liang Ji und Jim Piper

Im vorhergehenden Kapitel wurden die Grundlagen erläutert auf den das Iterative Verfahren von Liang und Jim Piper aufbaut. In jedem Iterationsschritt werden zuerst “Form” Punkte mittels das morphologische Skeletonverfahren gesucht. Danach werden diese Formpunkte mittels sogenannte J-Punkte verbunden, um die Homotopie zu erhalten.

3.1 Das Verfahren

3.1.1 Definitionen

- $I_i = I \ominus E_i$
  I ist das Orginalbild. In jedem Iterationsschritt i wird eine Hülle von den Objekten im Bild I_{i-1} entfernt. Dies wird solange wiederholt bis das Bild I_i leer ist.
- $S_i = I_i \backslash I_{i+1}$
  S_i stellt die i-te Hülle dar von das Objekt. Die Sequenz von Hüllen liefert die “Distanz Transformation (DT)” von I_1.
- $M_i = I_i \backslash (I_{i+1} \oplus D)$
  M-Punkte, oder Formpunkte. M_i sind die Formpunkte die aus die i-te Hülle gewonnen werden. Die Sequenz aller M_i stellt das eigentliche Skeleton dar. Die M-Punkte liefern aber nicht immer die gleiche Homotopie als das Orginalbild.
- J_i
  Homotopie erhaltende Punkte. J_i wird für jeden Iterationsschritt neu berechnet. Dieses verfahren ist im Abschnitt 3.2 erklärt.
- $X_i = X_{i-1} \cup M_i \cup J_i$
  X_0 ist leer. Nach durchlauf von alle Iterationen ist X_i das Skeleton von I_1.

3.1.2 Der Algorithmus

$i = 1$
`repeat while` $I_i \neq \emptyset$
$\{$
  `compute` I_i, S_i `and` M_i
  `compute` H_i
  $X_i = X_{i-1} \cup M_i \cup J_i$
  $i = i + 1$
$\}$
$n = i - 1$
`return` (X_n)

3.1.3 Die Kernels

Die Kernels D und E_i werden benutzt in den morphologischen Operationen. Es handelt sich in dem Fall von E_i dann um Sequenzen von Kernels, wobei die einzelnen Elemente in Abhängigkeit von ihren Durchmesser beschrieben werden können. Wichtig für die Geschwindigkeit der Algorithmus ist dabei, ob eine Sequenz von Kernels trennbar ist oder nicht (siehe auch Abschnitt 2.2.4). Eine Sequenz von Kernels ist trennbar, wenn man das nächste Element durch eine Dilatation des vorgehenden Elements mit dem ersten Kernel in der Sequenz berechnen kann ($E_{i+1} = E_i \oplus E_1$). Die drei Sequenzen die im Rahmen dieser Arbeit benutzt wurden, werden in den Abilldungen 3.1 - 3.3 kurz vorgestellt. In Prinzip funktioniert der Algorithmus mit jeder Sequenz von Kernels, die die Bedingung $E_i \oplus D \subseteq E_{i+1}$ entspricht. Im Fall einer trennbaren Sequenz E_i ist diese Bedingung immer erfüllt wenn $D \subseteq E_1$ ist [Ji und Piper, 1992].

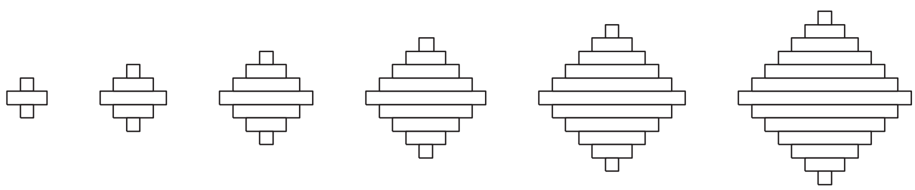

Abbildung 3.1: Der Cityblock-kernel

3.2 Die J-Punkte

J-Punkte dienen dazu die Homotopie eines Bildes zu erhalten. Sie sollten eingefügt werden wenn sonst der Zusammenhang zwischen verschiedenen Regionen eines Objektes wegen die

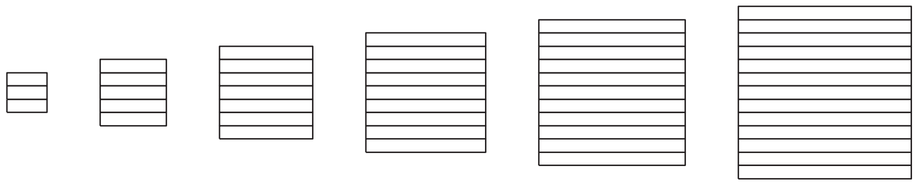

Abbildung 3.2: Der Chessboard-kernel

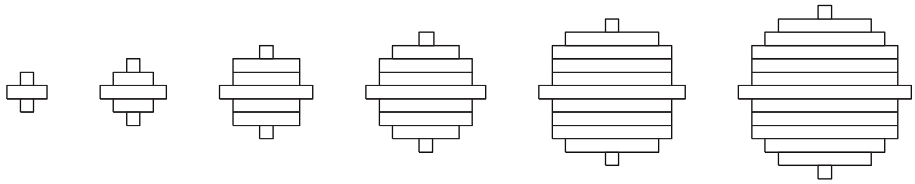

Abbildung 3.3: Der eukidische Kernel

Erosion verloren gehen würde. Um die Verbindung wieder herzustellen werden sowohl die Skeletonpunkte die bereits gefunden sind, X_{i-1}, als auch die aktuelle Hülle S_i in betracht gezogen.
Es sei $p \in S_i \backslash M_i$. Also liegt p auf die aktuelle Hülle, aber ohne die in diesen Iterationsschritt gefundenen Formpunkte. Weiter sei $N_p = N_8(p) \cap (S_i \cup X_{i-1})$. Das heißt, N_p sind die 8-Nachbarn von p, die zu den Formpunkten, oder zur aktuellen Hülle gehören. Jetzt ist p ein J-Punkt wenn gilt: $\exists a, b, c \in N_p$ so daß $a \notin N_4(b), b \notin N_4(c), c \notin N_4(a)$. Wenn es also drei Punkte in N_p gibt die nicht 4-benachbart sind, dann ist p ein J-Punkt.
Dieses Verfahren liefert manchmal redundante Punkte, das heißt Punkte die, streng genommen, nicht für die Homotopie-erhaltung notwendig sind. Diese Punkte ändern aber nicht die Form des Skeletons, nur die Breite.

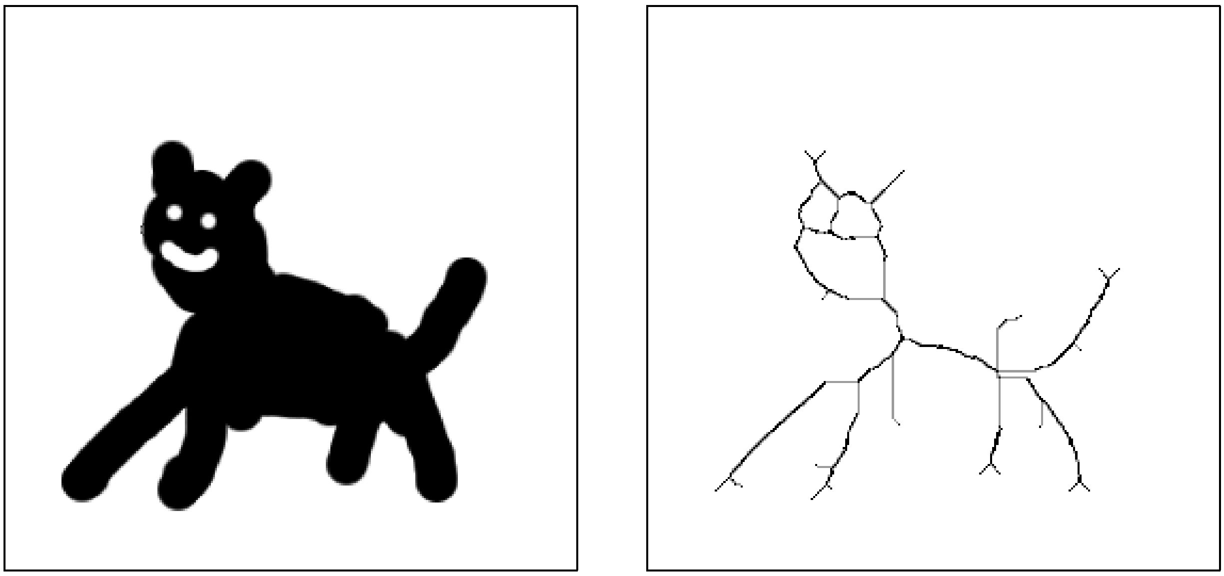

Abbildung 3.4: Ein Objekt und sein Skeleton: Die Formpunkte sind schwarz, die J-Punkte grau.

4 Implementierung

Der Algorithmus nach Ji und Piper wurde als Teil der LTILIB-Bibliothek des Lehrstuhls für Technische Informatik objektorientiert in C++ implementiert. Die Ausführung als Funktor mit einem integrierten Satz an Parametern ermöglicht eine Anpassung der Berechnung an verschiedene Anwendungsfälle und Eingabedaten.

4.1 Flußdiagramm

Das Flußdiagramm in Abbildung 4.1 stellt schematisch den Ablauf des Algorithmus dar, wie er in der *apply()*-Methode der Klasse `lti::skeleton` implementiert ist.

4.2 Schnittstelle

4.2.1 Eingabedaten

Als Eingabe wird ein Bild das ein oder mehrere Objekte (siehe Abschnitt 2.1.1) enthält erwartet. Die Eingabedaten müssen in Form von 1-Byte-Matrizen (`lti::channel8`) vorliegen. Das Bild wird als ein Binärbild verarbeitet. Das heißt das Pixeln mit den Wert 0 als Hintergrundpixeln betrachtet werden. Von Pixeln mit einem Wert unterschiedlich von 0 wird angenommen, daß sie zu einem Objekt gehören.

4.2.2 Ausgabedaten

Das Ausgabedatum ist ein Bild in der Form von einem 1-Byte-Matrix (`lti::channel8`). Das Bild enthält die Skeletons die zur Objekte aus das Eingabebild gehören.

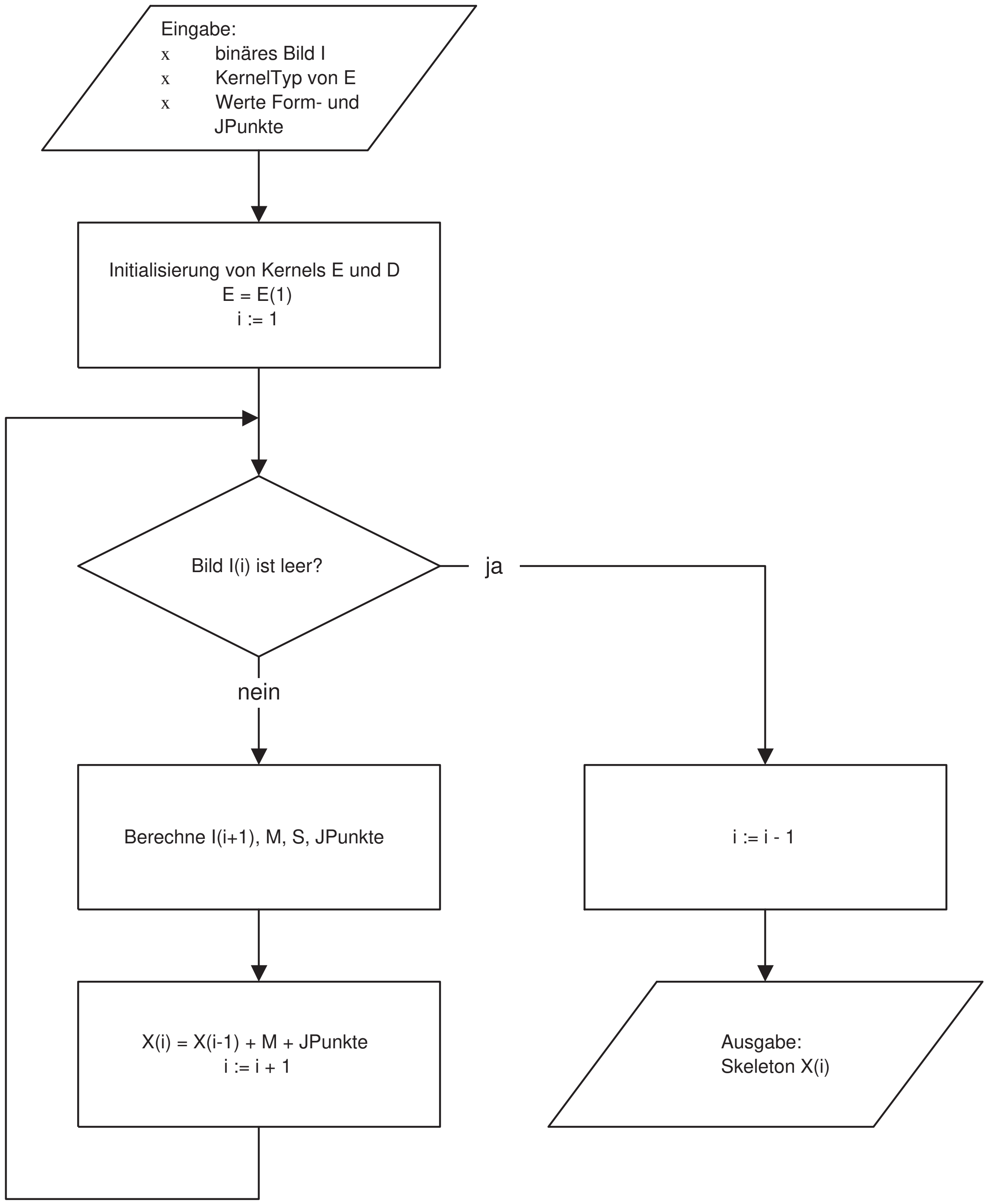

Abbildung 4.1: Flußdiagramm der Prägnanzberechnung

4.2.3 Die Parameter

Die eingebettete Parameter-Klasse `lti::skeleton::parameters` hat folgende Datenfelder:

```
enum eKernelType {
  CityBlock,
  ChessBoard,
  Euclidean
};
eKernelType kernelType;
int formPointsValue;
int jPointsValue;
```

4.2.3.1 Der Kerneltyp

In die Implementation des Algorithmus die im Rahmen dieser Arbeit geschrieben wurde, wird für den Dilatationskernel D immer ein Cityblockkernel mit Durchmesser 3 eingesetzt. Für den Erosionskernel E_i sind drei verschiedene Sequenzen von Kernels verfügbar: CityBlock, Chessboard und euklidischer Kernels (siehe auch Abschnitt 3.1.3). Über den Parameter `eKernelType kernelType` wird angegeben welcher Kernel benutzt werden sollte.

4.2.3.2 Darstellung der Formpunkte und J-Punkte

Über die Parameter `int formPointsValue` und `int jPointsValue` wird angegeben welcher Grauwert die Formpunkte und J-Punkte in das Ausgabebild haben werden. Standard sind beide auf den Maximalwert 255 eingestellt.

4.2.4 Die apply-Methoden

Die apply-Methoden sind die Schnittstellen zur Benutzung des Algorithmus. Es stehen zwei Funktionen für 8 bit integer-Matrizen (`lti::channel8`) zur Verfügung.

- `channel8& apply(channel8& srcdest) const;`
- `channel8& apply(channel8& src, channel8& dest) const;`

Die erste Variante ändert das Eingabebild in seine Skeletons. Die zweite Variante ändert das ursprüngliches Bild nicht. Dafür benötigt sie eine zweite Variable zur Ausgabe.

5 Zusammenfassung und Ausblick

Der in dieser Arbeit untersuchte Skeleton-Algorithmus von Liang Ji und Jim Piper hat sich als geeignetes Verfahren zur Erzeugung von Skeletons aus Binärbildern erwiesen. In einer weiteren Stufe kann die Skeleton-Darstellung zur Extraktion von Formmerkmalen Anwendung finden.

Vorteil dieser Algorithmus ist, daß bei geeigneter Wahl des Kernels (wenn die Sequenz der Erosionskernels trennbar ist) (Abschnitt 2.2.4 und 3.1.3) der Rechenaufwand zufriedenstellend ist. Die im Rahmen dieser Arbeit benutzte "trennbare" Sequenzen weisen aber eine Rotationsabhängigkeit auf. Das kann ein Nachteil sein. Der euklidische Kernel weist zwar keine Rotationsabhängigkeit auf, ist aber nicht trennbar und deswegen bei seinem Einsatz zur längeren Rechenzeiten führt.

In das Verfahren beschrieben durch [Ji und Piper, 1992] wird in jedem Iterationsschritt die redundanten J-Punkte entfernt. Da das Verfahren sowieso keine 1-Pixel breiten Skeletons liefert scheint es effektiver um die Entfernung redundanter Punkte in einem Schritt am Ende durchzuführen.

Durch eine Parallelisierung des Algorithmus kann, besonders auf einem Mehrprozessorsystem, die Geschwindigkeit weiter erhöht werden. Ein Einsatz im AXIOM-Projekt erscheint gerechtfertigt, wenn man die Eigenschaften des Verfahrens in Betracht zieht.

Literaturverzeichnis

[Ji und Piper, 1992]
Ji, Liang und Piper, Jim
(1992). *Fast Homotopy-Preserving Skeletons Using Mathematical Morphology IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 6 june 1992*, pp. 653–664.

[Gonzales und Woods, 1992]
Gonzales, Rafael C. und Woods, Richard E.
(1992). *Digital Image Processing*. Addison-Wesley publishing company

[Hassan und Karam, 2000]
Hassan, Yassin M.Y. und Karam, Lina J.
(2000). *Morphological Reversible Contour Representation IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 3 march 2000*, pp. 227–239.

[Sonka, Hlavac und Boyle, 1998]
Sonka, Milan, Hlavac, Vaclav und Boyle, Roger
(1998). *Image Processing, Analysis, and Machine Vision*. Brooks/Cole Publishing Company